An application for solving minimization problems using the Harmony search algorithm

Kilichev Dusmurod[1] and Zong Woo Geem[2*]

1 Department of IT Convergence Engineering, Gachon University, Seongnam, South Korea (dusmurod@gachon.ac.kr)
2 Department of Smart City & Energy, Gachon University, Seongnam, South Korea (geem@gachon.ac.kr)

* Corresponding Author

Abstract

The Harmony search algorithm has great accuracy and convenience in finding optimal solutions to computationally difficult optimization functions. A number of studies show that this algorithm has several innovative aspects in its operation that encourage its use in various fields such as engineering, telecommunications, robotics, construction, energy and healthcare. In this article, using the harmony search algorithm, various optimization problems aimed at minimization, including the basic structure of open source software developed for solving linear, non-linear, and discrete models, and its functionality shown in optimization examples.

Keywords

Harmony search algorithm, software of HS algorithm

Metadata

Nr	Code metadata description	*Please fill in this column*
C1	Current code version	v01.00
C2	Permanent link to code/repository used for this code version	https://github.com/TATU-hacker/Solver_of_minimization_problems
C3	Legal code license	GNU General Public License v3.0
C4	Code versioning system used	Git
C5	Software code languages, tools and services used	Python PyQt5 - version 5.15.0 Numpy - version 1.19.1 Matplotlib - version 3.3.1 os - Python Standard Library sys - Python Standard Library math - Python Standard Library random - Python Standard Library
C6	Support email for questions	geem@gachon.ac.kr

1. Motivation and significance

Geem et al. [1] developed a new harmony search (HS) meta-heuristic algorithm that was conceptualized using the musical process of searching for a perfect state of harmony. Compared to mathematical optimization algorithms, the HS algorithm imposes fewer mathematical requirements and does not require initial values for the decision variables.

Originally, applications where HS was first assessed as an effective meta-heuristic focused mainly on the design of water distribution networks [2], benchmark optimization [3], structural design [4] and vehicle routing problems [5], [6]. In 2004 a flowchart representation of HS was published in Lee and Geem [7] and since then several studies were devoted to the development of new HS variants and hybridizations with other meta-heuristic algorithms [8]. Since then, the activity around this algorithm has increased sharply, spanning its applicability to a very heterogeneous portfolio of application scenarios.

HS has so far been fully used as an algorithm to support various knowledge discovery methods in various disciplines and various application areas, such as feature selection, clustering, and planning.

The growing activity around this algorithm opens the door to interesting areas of future research, some of which are already being pursued by the scientific community. From a computational point of view, methods for efficient memory management and algorithm performance acceleration are among the next steps and directions in this field. In fact, many meta-heuristic approaches use the entire memory to obtain a new set of solutions. Therefore, parallelizing the code to minimize the computation time provides a research objective that is most relevant when solving very high-dimensional optimization problems (as mentioned above).

On the theoretical side, parameterization-free technique [9] should be further developed, because in practical applications, engineers and decision makers are ready to use HS without performing any parameterization procedure. In most cases, they want to "push a button" to get good solutions for their system, rather than seriously messing with the algorithm. Thus, how to provide end-users with more user-friendly parameter setting software is critical to the future success of this algorithm.

The developed software tool allows for much more effective matching search compared to other solvers. This is due to the use of algorithm parameters that evolve along with successive generations of vectors. HS algorithms consist of the following steps:

• Step 1. Initialization of the problem parameters and the algorithm.
• Step 2. Harmony memory initialization.
• Step 3. Improving a new harmony.
• Step 4. Update the harmony memory.
• Step 5. Checking the stop criterion.

The HMCR and PAR parameters introduced in Step 3 help the algorithm find globally and locally improved solutions, respectively.

PAR and bw in HS algorithm are very important parameters in fine-tuning of optimized solution vectors, and can be potentially useful in adjusting convergence rate of algorithm to optimal solution [10]. Therefore, in the development of this software, step 3 of the algorithm was modified as follows (Figure 1):

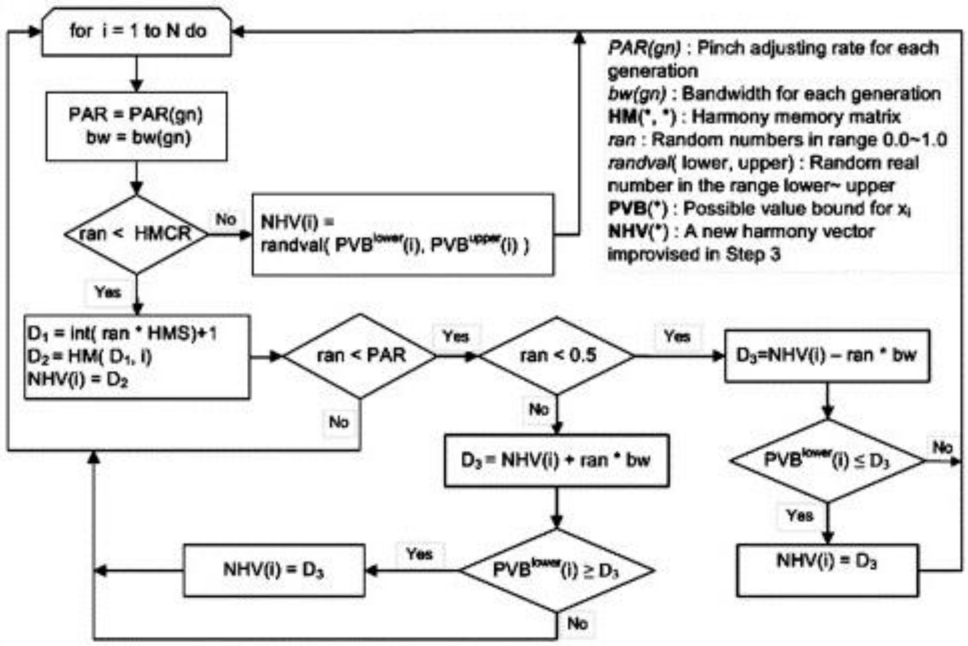

Figure 1. Improvisation harmony search algorithm without evolving HMCR parameter

2. Software description

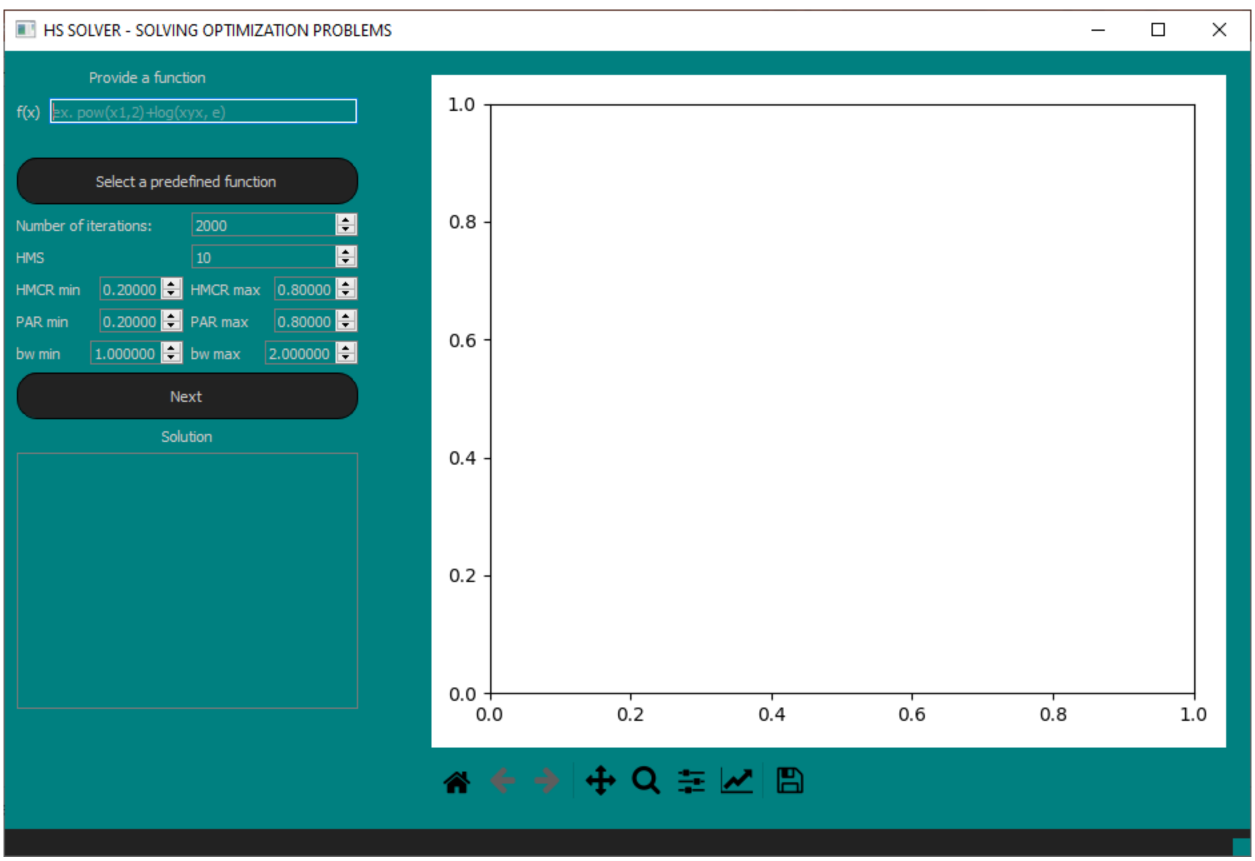

Figure 2. Software interface

The Python programming language was used in the development of this software tool. PyQt5 was used to create the Graphical User Interface part of the application. In the application used Matplotlib, a comprehensive library, to create interactive visualizations. NumPy library was used to perform operations with arrays.

The interface of the software is modern, simple and understandable. The sequence of actions is easy and convenient. The application interface is shown in Figure 2.

- The f(x) field is used to enter the desired function.
- The functions we introduce are collected in the functions.txt file. For our convenience, we can pre-include the necessary functions in the functions.txt file. The "Select a predefined function" button opens a window for selecting predefined functions (Figure 3).

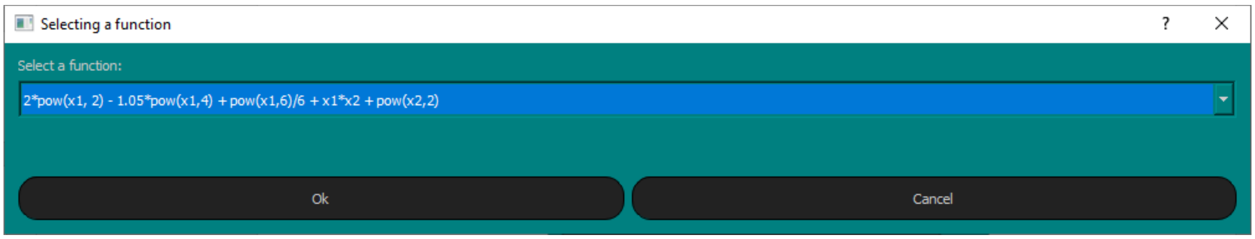

Figure 3. Select functions

- After entering or selecting a function, a message about defined variables will appear.
- In the next step, the parameters of the function are entered:
  Number of iterations - defines the number of iterations of the algorithm.
  HMS – specifies the size of the harmony memory.
  $HMCR_{min}$ and $HMCR_{max}$ – are responsible for the interval of evolution of the memory coverage index.
  PAR_{min} and PAR_{max} – are responsible for the interval of evolution of the "pitch" control indicator.
  bw_{min} and bw_{max} – are responsible for the range of harmony search bandwidth.
- If any parameter has been entered incorrectly, an appropriate message will appear and the possibility of going further will be blocked.
- After pressing the button Next, the window responsible for retrieving the ranges of values for the variables detected in the equation from the user will appear (Figure 4). If a parameter has been entered incorrectly, an appropriate message will be displayed.

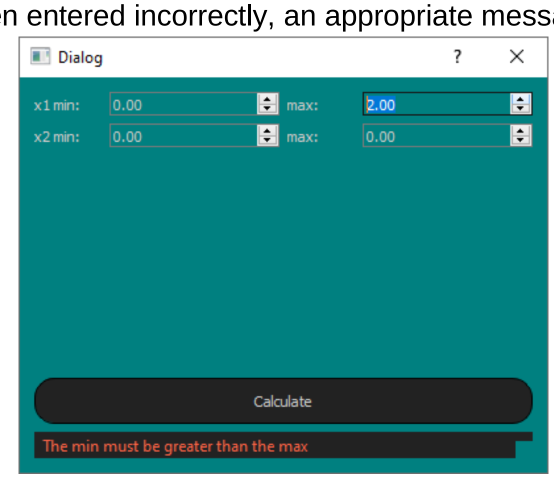

Figure 4. The range of values of variables

- After correct data input and calculation of the solution, the function contour plot and the next best points found by the algorithm will appear in the main window (Figure 5).
- In the solution field, the values of the variables and the value of the function at the found point will appear (Figure 5).

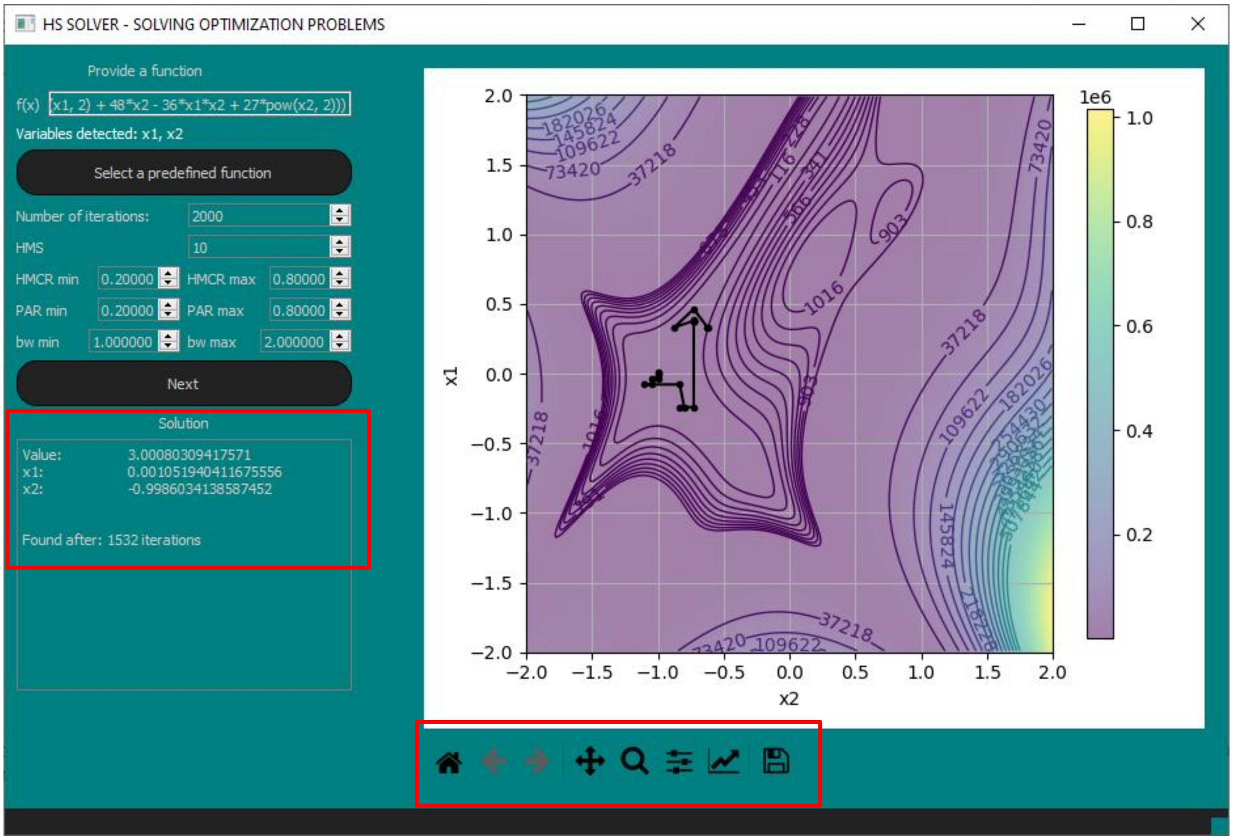

Figure 5. The function contour plot and the found points

- The chart has a Toolbar from the matplotlib package, which allows for basic operations on the chart, e.g. zooming and saving to a file.

3. Illustrative examples

We consider the following function:

$$f(\vec{x}) = \{1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\} \times$$
$$\times \{3 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\}$$

$$\min f(\vec{x}) = f(0, 1) = 3$$

For this function, an algorithm was run with various combinations of parameter values. These values for each test case are shown in the Table 1 below. The ranges of the variables are:

$$\begin{cases} -2 < x_1 < 2, \\ -2 < x_2 < 2. \end{cases}$$

Table 2. Values for each test case

ID	L	HMS	$HMCR$	PAR	bw	**Results**
1	100	5	0.2-0.8	0.2-0.8	0-1	69.5196
2	1000	5	0.2-0.8	0.2-0.8	0-1	3.4156
3	1000	100	0.2-0.8	0.2-0.8	0-1	5.9362
4	1000	100	0-0	0.2-0.8	0-1	4.8874
5	1000	5	0-0	0.2-0.8	0-1	168.451
6	1000	5	1-1	0.2-0.8	0-1	3.1735
7	1000	100	1-1	0.2-0.8	0-1	263.993
8	1000	10	1-1	0-0	0-1	3.679
9	1000	10	1-1	1-1	0-1	3.478
10	1000	10	0.2-0.8	0-0	0-1	3.052
11	1000	10	0.2-0.8	1-1	0-1	3.1479
12	10000	10	0.2-0.8	0.2-0.8	0-1	3.045
13	**10000**	**10**	**0.2-0.8**	**0.2-0.8**	**1-2**	**3.0008**

For each of the many calls to the first case, the result was different, so 100 iterations is not enough for further consideration (Case 1).
If the number of iterations is set to 1000, in most cases the program returns the minimum value close to the global minimum (Case 2).
After increasing the parameter HMS in case 3, it was noticed that the algorithm finds fewer new vectors and the results are less accurate.
For the parameter HMCR = 0, the change of any other parameter does not cause noticeable changes in the operation of the algorithm. The results in this case are not very precise (Case 4). This is due to the fact that for the parameter HMCR = 0, the parameters PAR and bw are not taken into account, because the algorithm does not reach the moment of their use. The parameter HMS is also irrelevant, because regardless of it, new points are selected completely randomly from the entire search range.
In case of setting the parameter HMCR = 1, it can be noticed that with a low value of the parameter HMS the algorithm does not find any solution (Case 5).
But if the value of HMS is increased, the algorithm starts to work better, because it finds values close to optimal (Case 6). This can be explained as follows: with HMCR = 1 all new points are generated taking into account the stored points. The algorithm in each generation can search for

new coordinates at a distance of bw from each of the stored coordinates. In this case, the greater number of remembered points HMS means that in the greater part of the domain the algorithm can search for new solutions.
In case 8, where the parameter HMCR=1 and PAR=0, it can be seen that for each call, only single points are found. This is related to the fact that the algorithm is forced to search for the most optimal solution only from initially drawn HMS points.
Setting the parameter PAR = 1 in the case of 9 did not cause a significant change in the results.
In the case of 10, when the parameter $HMCR$ evolves between 0.2 and 0.8, an improvement can be noticed compared to HMCR = 0. This is due to the fact that new coordinates are not only drawn from the entire domain, but also the best remembered solutions.
Case 11 shows that by adjusting the pitch (when $PAR > 0$), more accurate results are obtained than without, and even better results are obtained by setting the parameter PAR to the range (0.2 - 0.8), as in the case of 12.
After a thorough analysis of case 13, it can be seen that after increasing the parameter bw, the algorithm went through all local minima, but finally found the global minimum, although with less accuracy than for the parameter bw with a lower value (Figure 6).

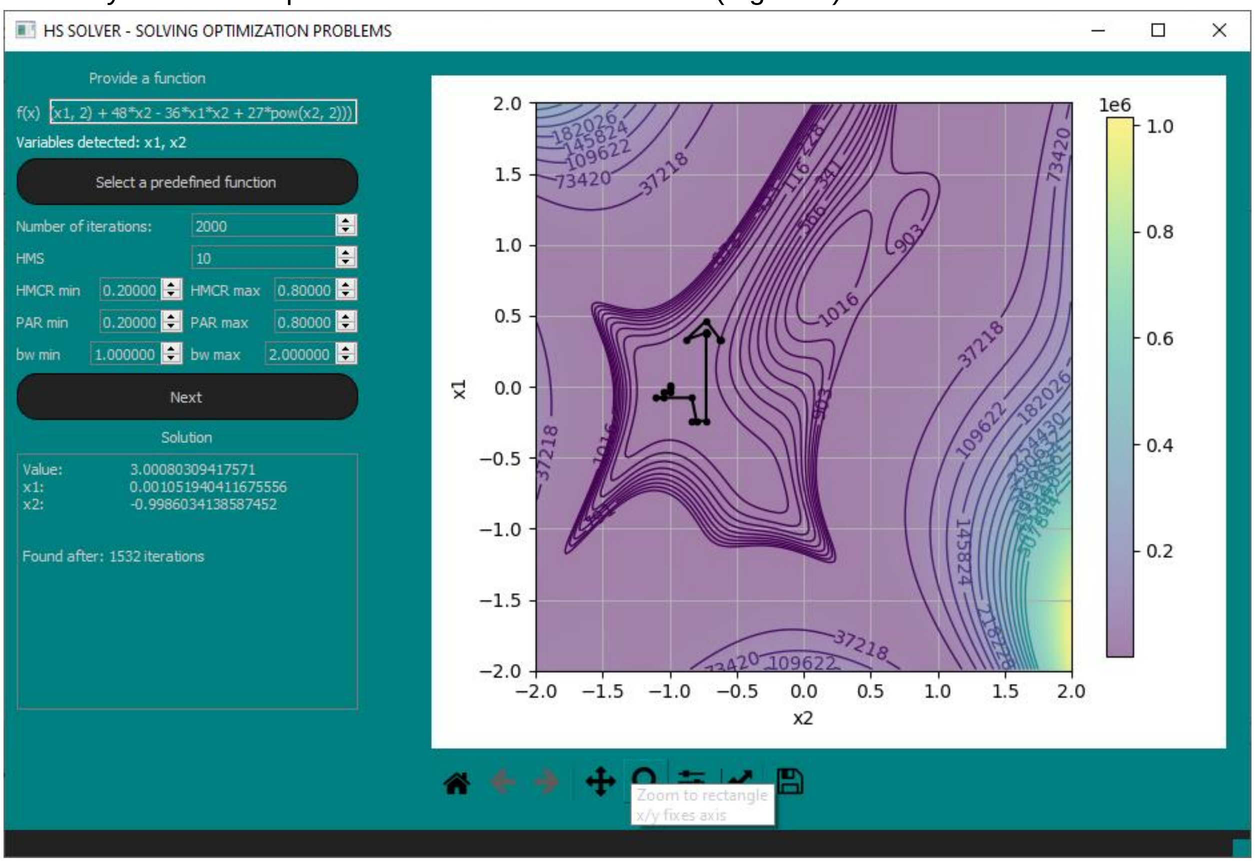

Figure 6. The global minimum

4. Impact

Let us consider the use of this application in engineering science problems, where this solver is reported to be a viable alternative to other traditional optimization methods. Thus, it is widely used in complex optimization problems that appear in many engineering problems, such as heat exchanger design optimization, steel, electronic, mechanical, telecommunication, construction and engineering structure problems, etc.

For example, in the following cases, the use of this application, created in the problems of finding the optimal minimum of the problem, is very convenient, allows to achieve high accuracy and efficiency:

- Typical steel engineering problems include structural design optimization problems. Such problems usually require the selection of steel members for its beams and columns according to criteria that meet the frame's serviceability and strength requirements, while minimizing the material costs of the frame. In addition to the budget constraint, this choice is usually made in such a way that the steel frame has a minimum weight.
- Shell and tube heat exchangers (STHX) are the most widely used heat exchangers in the process industry due to their relatively simple manufacturing and adaptability to various operating conditions. STHX design, including thermodynamic and fluid dynamic design, cost estimation and optimization, is a complex process involving design rules and empirical knowledge from various fields.
- Improving the energy efficiency and environmental performance of buildings is a major priority worldwide. New building regulations clearly focus on low-emission and energy-efficient designs. However, the optimal design of residential buildings must consider multiple and often competing objectives, such as optimizing energy consumption, reducing financial costs, and minimizing environmental impact.
- Several studies have focused on the design of water distribution networks, in particular on the selection of optimal pipe diameters. Typically, there is one fixed-pressure supply node and many demand nodes, which form the structure of the connection network. Both heights and distances between nodes (pipe lengths) are shown. Therefore, the goal of these studies is to choose the diameter for each pipe segment that minimizes the total cost of the water distribution network.
- Most of the work related to energy is focused on the problem of energy flow optimization, the goal of which is to determine the loads in megawatts that must be supplied by certain nodes or buses of the transmission system in a way that requires minimum costs.

5. Conclusions

During the development of this software tool, the following conclusions were drawn based on the application of the HS algorithm, conducting test results and design assumptions:

- Parameter L - number of iterations - affects the accuracy of the result to the greatest extent, provided that the other parameters are selected so as not to block the functionality of the algorithm. A large number of iterations increases the computation time, but also the accuracy of the result obtained, and ensures that the real optimum for a given range will be found.
- Parameter HMS - number of vectors stored in memory. Its high value increases the computation time. For values of $HMCR$ close to 1, the parameter HMS has a large impact on the quality, and for values of $HMCR$ close to 0 HMS it does not have a large impact on the quality of the result.
- Parameter $HMCR$ - memory consideration index - influences the real determinism of obtaining the optimal result. The higher it is, the more the algorithm focuses on searching near the minima found, but it can get bogged down in the local minimum. This determines the necessity of increasing the parameter HMS together with increasing the parameter $HMCR$, which to some extent may prevent the algorithm from being stopped at a local minimum.

- Parameter PAR - an indicator of adjusting the value of a vector element. It is important for high values of $HMCR$, because it allows searching for new values of vector elements in the band instead of in the line, which allows you to get closer to the optimum. It is also closely related to the bw parameter, because for high values of bw it weakens the propensity of the algorithm with a high $HMCR$ index to get stuck in local optimas, and lowering the value of bw increases the accuracy of the result.
- Variable value ranges - The larger the ranges, the less accurate the result. Additionally, by changing ranges around the optims, you can force local optims to be found.

Acknowledgements

References

[1] Geem ZW, Kim JH, Loganathan GV. A new heuristic optimization algorithm: harmony search, Simulation 2001; 76(2):60–68. https://doi.org/10.1177/003754970107600201

[2] Z.W. Geem. Optimal cost design of water distribution networks using harmony search. Eng. Optim., 38 (2006), pp. 259-280

[3] Li, H.Q., Li, L., 2007. A novel hybrid particle swarm optimization algorithm combined with harmony search for high dimensional optimization problems. In: International Conference on Intelligent Pervasive Computing Korea.

[4] K.S. Lee, Z.W. Geem, S.H. Lee. The harmony search heuristic for discrete structural optimization. Eng. Optim., 37 (2005), pp. 663-684

[5] Geem, Z.W., 2005. School bus routing using harmony search. In: Genetic and Evolutionary Computation Conference, Washington DC.

[6] Z.W. Geem, C.-L. Tseng, Y. Park. Harmony search for generalized orienteering problem: best touring in China. Lect. Notes Comput. Sci., 3612 (2005), pp. 741-750

[7] K.S. Lee, Z.W. Geem. A new structural optimization method based on the harmony search algorithm. Comput. Struct., 82 (9–10) (2004), pp. 781-798

[8] L. Li, S.C. Chi, G. Lin. Genetic algorithm incorporated with harmony procedure and its application to searching of non-circular critical slip surface in soil slopes. J. Hydraul. Eng., 36 (2005), pp. 91-918

[9] Z.W. Geem, K.B. Sim. Parameter-setting-free harmony search algorithm. Appl. Math. Comput., 217 (8) (2010), pp. 3881-3889

[10] M. Mahdavi, M. Fesanghary, E. Damangir. An improved harmony search algorithm for solving optimization problems. Appl. Math. Comput., 188 (2) (2007), pp. 1567-1579